MODULE *Handshake*

$$\textsc{constant}\ \mathit{clients}$$

$$\begin{array}{ll}\textsc{variable} & \mathit{clientState},\\ & \mathit{serverState},\\ & \mathit{clientsAcknowledged},\\ & \mathit{msgs}\end{array}$$

$$\begin{array}{lll}\mathit{Messages} & \triangleq & [\mathit{type} : \{\text{“syn”}\},\ \mathit{client} : \mathit{clients}]\\ & & \cup\,[\mathit{type} : \{\text{“syn+ack”}\},\ \mathit{client} : \mathit{clients}]\\ & & \cup\,[\mathit{type} : \{\text{“ack”}\},\ \mathit{client} : \mathit{clients}]\end{array}$$

syn message from a client
syn+ack message from server to a client *ack* message from a client to server

$$\begin{array}{lll}\mathit{HSTypeOK} & \triangleq & \land\ \mathit{clientState}\ \in [\mathit{clients} \rightarrow \{\text{“working”},\ \text{“attempting”},\ \text{“acknowledged”}\}]\\ & & \land\ \mathit{serverState} \in \{\text{“working”},\ \text{“blocked”}\}\\ & & \land\ \mathit{msgs} \subseteq \mathit{Messages}\\ & & \land\ \mathit{clientsAcknowledged} \subseteq \mathit{clients}\end{array}$$

$$\begin{array}{lll}\mathit{HSInit} & \triangleq & \land\ \mathit{serverState} = \text{“working”}\\ & & \land\ \mathit{clientState}\ = [r \in \mathit{clients} \mapsto \text{“working”}]\\ & & \land\ \mathit{msgs} = \{\}\\ & & \land\ \mathit{clientsAcknowledged} = \{\}\end{array}$$

$$\begin{array}{lll}\mathit{ClientInitiates}(r) & \triangleq & \land\ \mathit{clientState}[r] = \text{“working”}\\ & & \land\ \mathit{serverState} = \text{“working”}\\ & & \land\ \mathit{clientState}' = [\mathit{clientState}\ \textsc{except}\ ![r] = \text{“attempting”}]\\ & & \land\ \mathit{msgs}' = \mathit{msgs} \cup \{[\mathit{type} \mapsto \text{“syn”},\ \mathit{client} \mapsto r]\}\\ & & \land\ \textsc{unchanged}\ \langle \mathit{serverState},\ \mathit{clientsAcknowledged}\rangle\end{array}$$

$$\begin{array}{lll}\mathit{ServerResponds}(r) & \triangleq & \land\ \mathit{serverState} = \text{“working”}\\ & & \land\ \mathit{clientState}[r] = \text{“attempting”}\\ & & \land\ [\mathit{type} \mapsto \text{“syn”},\ \mathit{client} \mapsto r] \in \mathit{msgs}\\ & & \land\ \mathit{msgs}' = \mathit{msgs} \cup \{[\mathit{type} \mapsto \text{“syn+ack”},\ \mathit{client} \mapsto r]\}\\ & & \land\ \textsc{unchanged}\ \langle \mathit{serverState},\ \mathit{clientState},\ \mathit{clientsAcknowledged}\rangle\end{array}$$

$$\begin{array}{lll}\mathit{ClientConfirms}(r) & \triangleq & \land\ \mathit{clientState}[r] = \text{“attempting”}\\ & & \land\ \mathit{serverState} = \text{“working”}\\ & & \land\ [\mathit{type} \mapsto \text{“syn+ack”},\ \mathit{client} \mapsto r] \in \mathit{msgs}\\ & & \land\ \mathit{msgs}' = \mathit{msgs} \cup \{[\mathit{type} \mapsto \text{“ack”},\ \mathit{client} \mapsto r]\}\\ & & \land\ \mathit{clientState}' = [\mathit{clientState}\ \textsc{except}\ ![r] = \text{“acknowledged”}]\\ & & \land\ \textsc{unchanged}\ \langle \mathit{serverState},\ \mathit{clientsAcknowledged}\rangle\end{array}$$

$$\begin{array}{lll}\mathit{ServerConfirms}(r) & \triangleq & \land\ \mathit{serverState} = \text{“working”}\\ & & \land\ \mathit{clientState}[r] = \text{“acknowledged”}\end{array}$$

$$\begin{array}{l}
\land [type \mapsto \text{“ack”},\ client \mapsto r] \in msgs \\
\land clientsAcknowledged' = clientsAcknowledged \cup \{r\} \\
\land \textsc{unchanged}\ \langle serverState,\ clientState,\ msgs \rangle
\end{array}$$

$ServerBlocked \triangleq$

Server may be unable to service requests *i.e.* it cannot perform handshaking temporarily. This may occur if it reaches max capacity or if it crashes. In this case, it halts all handshaking,shown by the *msgs* set being emptied. It does not require acknowledged clients to perform handshaking again.

$$\begin{array}{l}
\land serverState = \text{“working”} \\
\land serverState' = \text{“blocked”} \\
\land msgs' = \{\} \\
\land clientState' = [r \in \textsc{domain}\ clientState \mapsto \\
\qquad\qquad \textsc{if}\ clientState[r] = \text{“attempting”}\ \textsc{then}\ \text{“working”} \\
\qquad\qquad\ \ \textsc{else}\ \ clientState[r]] \\
\land \textsc{unchanged}\ \langle clientsAcknowledged \rangle
\end{array}$$

$$\begin{array}{ll}
ServerUnblocked \triangleq & \land serverState = \text{“blocked”} \\
& \land serverState' = \text{“working”} \\
& \land \textsc{unchanged}\ \langle clientState,\ msgs,\ clientsAcknowledged \rangle
\end{array}$$

$$\begin{array}{lll}
HSNext \triangleq & \lor ServerBlocked & \\
& \lor ServerUnblocked & \\
& \lor \exists\, r \in clients : & \lor ClientInitiates(r) \\
& & \lor ServerResponds(r) \\
& & \lor ClientConfirms(r) \\
& & \lor ServerConfirms(r)
\end{array}$$

$$HSCorrectness \triangleq \ \land \forall\, r \in clientsAcknowledged : clientState[r] = \text{“acknowledged”}$$

A client that has been acknowledged by server has to be in the acknowledged state

$$\begin{array}{ll}
HSSpec \triangleq & \land HSInit \\
& \land \Box[HSNext]_{\langle clientState,\ serverState,\ clientsAcknowledged,\ msgs \rangle} \\
& \land \Box HSTypeOK \\
& \land \Box HSCorrectness
\end{array}$$

* Modification History

* Last modified *Fri Jan* 10 10:50:26 *IST* 2020 by *vedant*

* Created *Wed Jan* 08 14:49:34 *IST* 2020 by *vedant*